Drones

Universidad de Alicante - Grado en Ingeniería Robótica

Daniel Frau Alfaro, Pablo García Gómez y José Ángel Martín González

***Palabras clave*—DSA, dinámica, SMC, Backstepping, PID, SLAM**

I. Introducción

Los vehículos aéreos no tripulados, más conocidos como drones, se están volviendo más comunes en muchos ámbitos como reconocimiento, operaciones de rescate, ocio o agricultura... Son muy usados debido a las ventajas que presentan, ya que al ser pequeños permiten una gran maniobrabilidad lo que posibilita realizar trayectorias complejas.

Estos dispositivos tienen 6 grados de libertad y tantas acciones de control como rotores tenga ya que estos dependen de la forma.

II. Hardware

A. *Tipos de drones según su estructura*

1) ***Ala fija****:* Como su nombre indica, en estos drones, las alas son fijas, es decir, que no pueden variar su posición. Estos drones suelen tener una forma similar a un avión, y pueden ser propulsados por motores eléctricos o a combustión. Son los que aprovechan la aerodinámica para su elevación ya que son las alas las que generan esa fuerza de sustentación en el aire.

2) ***Ala rotatoria****:* Son más conocidos en el mercado actual por la versatilidad, estos logran la sustentación por medio de las hélices situadas en cada uno de sus brazos. Son capaces de mantenerse sobrevolando un mismo punto y de aterrizar y despegar de manera vertical

B. *Tipos de drones según el número de hélices*

1) ***Tricópteros****:* Un tricóptero es un dron que tiene tres hélices o brazos. Es menos común que otro tipo de drones, pero tiene la ventaja de tener una hélice adicional que le permite mantener un mejor control en vuelos al aire libre y en condiciones de viento moderado. También, su motor adicional ayuda a compensar cualquier fallo mecánico en los otros motores.

2) ***Cuadricópteros****:* Un cuadricóptero es un dron que tiene cuatro hélices o brazos. Es el tipo más común de dron multirotor y es muy versátil. Puede ser utilizado para una amplia variedad de aplicaciones, como el vuelo de fotografía y vídeo aéreo, vigilancia, inspección, etc...

3) ***Hexacópteros****:* Un hexacóptero es un dron que tiene seis hélices o brazos. Es muy estable y maniobrable, lo que lo hace adecuado para vuelos en condiciones de viento fuerte y aplicaciones que requieren una gran estabilidad y precisión, como la fotografía aérea profesional y grabaciones en cine y televisión.

4) ***Octocópteros****:* Un octocóptero es un dron que tiene ocho hélices o brazos. Es muy estable y maniobrable, y es capaz de transportar cargas más pesadas que los drones de menos hélices. Por esta razón, son utilizados para aplicaciones que requieren una gran fuerza de elevación.

5) ***Coaxiales****:* Un dron coaxial es un tipo especial de dron multicóptero que tiene normalmente dos hélices que giran en direcciones opuestas en el mismo eje. Esto le proporciona una mayor estabilidad y maniobrabilidad, haciéndolo muy fácil de pilotar. Estos drones son populares en aplicaciones de entretenimiento, vuelos en interiores, y también para aprendizaje y formación en vuelo.

III. Modelos de drones

A. *Quantum Vector*

Es un dron de ala fija que tiene dos ramas en su aplicación, la primera es para realizar tareas de espionaje, aunque tiene una variante para realizar trabajos de carga.

Para su primer propósito, el dron está equipado con una cámara capaz de realizar hasta 20 aumentos en HD, además posee un sensor térmico de baja luminosidad.

En su despegue puede levantar hasta 7.4 kg, mide 1.63 x 2.8 metros y llega hasta 72 km en velocidad crucero.

Fig. 1. Quantum Vector

B. *DJI Mavic Pro*

Es un cuadricóptero capaz de plegarse para su transporte, de tamaño reducido, que además posee una cámara para realizar fotografías o vídeos. Llega a alcanzar una velocidad de 64.8 Km/h

Este dron posee varias tecnologías en su software:

- Detección de lugares en los que está prohibido volar
- Aterrizaje y despegue asistido
- Detección de obstáculos
- Botón del pánico
- Sistema de detección de personas y control por gestos

Fig. 2. DJI Mavic Pro

C. DJI Spreading Wings S1000+

Es un octocóptero construido en fibra de carbono que pesa 4.4 Kg, puede levantar hasta 11 Kg en el despegue. Tiene un gimbal que se monta bajo el marco en un soporte específicamente diseñado que se combina con un tren de aterrizaje retráctil.

Fig. 3. DJI Spreading Wings S1000+

IV. Cinemática de drones

Para la cinemática de un dron se deben definir los diferentes sistema de referencias, de manera que se relaciona el origen de coordenadas del robot con el del mundo. En el modelado, el dron se representa como una cruz con motores en los extremos.

El vector $r = [x\ y\ z]^T$ representa la posición del vehículo en el mundo, mientras que la orientación del mismo se expresa en *roll, pitch, yaw* (ϕ, θ, ψ). Así, la matriz de rotación entre la orientación del mundo y la del dron quedaría de la siguiente manera:

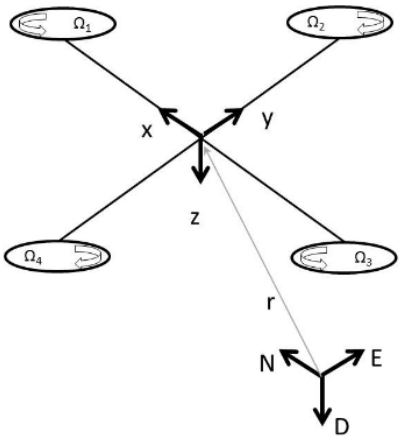

Fig. 4. Sistemas de referencia del mundo y del dron

$$R = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\theta c\psi & c\phi s\theta s\psi - s\theta c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \quad (1)$$

El c y s indican coseno y seno. Esta matriz se obtiene al multiplicar las matrices de rotación de todos los ejes. Con la ecuación anterior se pueden trasladar las magnitudes entre los dos sistemas de referencia.

Por otro lado, para relacionar las velocidades se tiene la siguiente matriz:

$$R_r = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & c(\theta) & s(\phi)c(\theta) \\ 0 & -s(\phi) & c(\phi)c(\theta) \end{bmatrix} \quad (2)$$

Así, si el robot se encuentra horizontal al suelo, los sistemas de referencia del dron y el mundo coincidirán, siendo la matriz R_r la matriz identidad. [3]

V. Dinámica de drones

Los drones son vehículos altamente inestables y no lineales, por lo que antes de plantear cualquier tipo de controlador se debe tener en cuenta la dinámica del sistema. Este tipo de robots presentan elementos como rotores o perturbaciones como el aleteo de las turbinas, que son elementos a tener en cuenta en el modelo con tal de que sea preciso y acorde a la realidad. A continuación se va a exponer la dinámica de un robot volador de cuatro hélices o *quadrotor*. [3] [4]

La dinámica de un UAV o dron expresa el comportamiento real del mismo a través de ecuaciones matemáticas diferenciales que, como se dijo anteriormente, resultan no lineales.

A. *Ecuaciones de movimiento*

Las ecuaciones de movimiento describen los movimientos en cada uno de los ejes y orientaciones del robot, transformando las magnitudes en el espacio articular, que en este caso son las velocidades de los rotores, a desplazamientos en el espacio cartesiano, con XYZ y *roll, pitch, yaw*.

De esta manera, primero se definen las ecuaciones de movimiento en orientación basadas en las ecuaciones de la dinámica de Newton - Euler:

$$J\dot{w} + w \times Jw + M_G = M_B \tag{3}$$

Donde J es la matriz de inercia, w es la velocidad angular, M_G los momentos producidos por el rotor y M_B son los momentos sobre el dron. Así, se puede expresar esa misma ecuación sustituyendo la inercia del motor como:

$$J\dot{w} + w \times Jw + w \times [0\,0\,J_r\Omega_r]^T = M_B \tag{4}$$

Donde J_r es la inercia del motor y Ω_r es la velocidad total del cuerpo expresada como la suma de todas sus velocidades considerando positiva la dirección positiva del eje Z del cuerpo del robot $\Omega_r = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$.

Un fenómeno característico de los drones es la fuerza que se produce con el rozamiento con el aire y el debido al giro de las hélices. Así, las expresiones de la fuerza aerodinámica y la generada por el momento aerodinámico resultaría como:

$$F_i = \frac{1}{2}\rho A C_T r^2 \Omega_i^2 \tag{5}$$

$$M_i = \frac{1}{2}\rho A C_D r^2 \Omega_i^2 \tag{6}$$

Donde ρ es la densidad del aire, A es el área de las hélices, C_T y C_D son los coeficientes aerodinámicos, r_b es el radio de la hélice y Ω_i es la velocidad del i-ésimo rotor. Teniendo en cuenta las limitaciones de diseño, las expresiones se pueden reescribir a:

$$F_i = K_f \Omega_i^2 \tag{7}$$

$$M_i = K_M \Omega_i^2 \tag{8}$$

Donde las ganancias K se presentan como constantes respecto a las variables anteriores, pues los valores no cambian en entornos relativamente pequeños.

De esta manera, se puede definir el momento total sobre el dron M_B, sabiendo que cada motor genera una fuerza hacia arriba y un momento en la dirección contrario al giro de los rotores. La siguiente figura muestra el diagrama de esfuerzos:

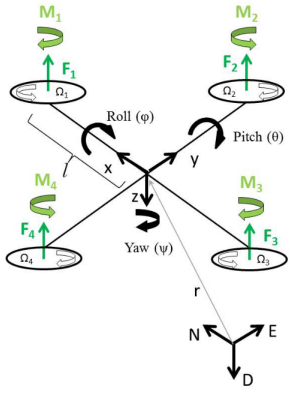

Fig. 5. Esfuerzos sobre el dron debido a las fuerzas y momentos de los motores

De la anterior figura se puede extraer los momentos en el sistema de referencia del dron:

$$\begin{aligned} M_x &= -F_2 l + F_4 l \\ &= -(K_f\Omega_2^2)l(K_f\Omega_4^2)l \\ &= lK_f(-\Omega_2^2 + \Omega_4^2) \end{aligned} \tag{9}$$

$$\begin{aligned} M_y &= -F_1 l + F_3 l \\ &= -(K_f\Omega_1^2)l(K_f\Omega_3^2)l \\ &= lK_f(-\Omega_1^2 + \Omega_3^2) \end{aligned} \tag{10}$$

$$\begin{aligned} M_z &= M_1 - M_2 + M_3 - M_4 \\ &= (K_M\Omega_1^2) - (K_M\Omega_2^2) + (K_M\Omega_3^2 - K_M\Omega_4^2) \\ &= K_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{aligned} \tag{11}$$

Expresándolo en forma matricial:

$$M_B = \begin{bmatrix} lK_f(-\Omega_2^2 + \Omega_4^2) \\ lK_f(-\Omega_1^2 + \Omega_3^2) \\ K_M(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{bmatrix} \tag{12}$$

Donde l es la distancia entre el centro de referencia con cada uno de los motores, que será el mismo para cada uno de los brazos.

En cuanto a la fuerza sobre el dron, se define la segunda ley de Newton:

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + RF_B \tag{13}$$

Donde m es la masa del robot, r es la posición del robot en el sistema de referencia del mundo, g es la gravedad del planeta donde se vuele el dron, R es la matriz de rotación de (1) y F_B es la fuerza no asociada a la gravedad en el dron. A la hora de definir esta última magnitud se supone una situación ideal en la que el dron se encuentra paralelo al suelo, por lo que solo las fuerzas de los rotores es la única que actúa sobre el mismo. Así:

$$F_B = \begin{bmatrix} 0 \\ 0 \\ -K_f(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2+) \end{bmatrix} \tag{14}$$

B. Modelo en el espacio de estados

El espacio de estados es una manera intuitiva de representar las variables a controlar del sistema, estableciendo relaciones entre las mismas y las entradas al sistema. Además, se pueden definir las entradas de control al robot, por las cuales enviar comandos de movimiento.

De esta manera, se tiene una representación de estados con 12 estados a controlar:

$$X = [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9\ x_{10}\ x_{11}\ x_{12}]^T \tag{15}$$

$$X = [x\quad \dot{x}\quad y\quad \dot{y}\quad z\quad \dot{z}\quad \phi\quad \dot{\phi}\quad \theta\quad \dot{\theta}\quad \psi\quad \dot{\psi}]^T \tag{16}$$

Los estados son la posición en el espacio cartesiano del robot junto con las velocidades, y las posiciones angulares y sus respectivas velocidades angulares. Estos valores están en el sistema de referencia del mundo, por lo que sería necesario efectuar transformadas mediante matrices de transformación homogénea.

Luego, las variables de control se expresan siguiendo las siguientes ecuaciones:

$$U_1 = K_f(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \tag{17}$$

$$U_2 = K_f(-\Omega_2^2 + \Omega_4^2) \tag{18}$$

$$U_3 = K_f(-\Omega_1^2 + \Omega_3^2) \tag{19}$$

$$U_4 = K_M(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \tag{20}$$

Donde U_1 es la que controla la altitud del dron, pues tiene todas las velocidades de los cuatro motores, mientras que U_2, U_3, U_4 son responsables de las rotaciones en *roll, pitch, yaw* respectivamente, teniendo las variables controladas por separado. Expresándolo en forma matricial:

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} K_f & K_f & K_f & K_f \\ 0 & -K_f & 0 & K_f b \\ -K_f & 0 & K_f & 0 \\ -K_M & K_M & -K_M & K_M \end{bmatrix}^{-1} * \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{21}$$

Para realizar el control se necesita calcular las velocidades a partir de las señales de control, por lo que despejando se llega a las siguientes ecuaciones:

$$\Omega_1 = \sqrt{\frac{1}{4K_f}U_1 + \frac{1}{2K_f}U_3 + \frac{1}{4K_M}U_4} \tag{22}$$

$$\Omega_2 = \sqrt{\frac{1}{4K_f}U_1 - \frac{1}{2K_f}U_2 - \frac{1}{4K_M}U_4} \tag{23}$$

$$\Omega_3 = \sqrt{\frac{1}{4K_f}U_1 - \frac{1}{2K_f}U_3 + \frac{1}{4K_M}U_4} \tag{24}$$

$$\Omega_4 = \sqrt{\frac{1}{4K_f}U_1 + \frac{1}{2K_f}U_2 - \frac{1}{4K_M}U_4} \tag{25}$$

Una vez se tienen las ecuaciones para calcular las velocidades de los motores, se tiene el modelo *articular* del sistema. Por ello, se necesita pasar ese modelo al espacio cartesiano, de manera que el dron tenga modelados movimiento en *XYZ* así como en $\phi\theta\psi$.

Por ello, se igualan las variables de control ((18), (19), (20)) a los momentos totales ((12)):

$$M_B = \begin{bmatrix} lU_2 \\ lU_3 \\ U_4 \end{bmatrix} \tag{26}$$

Desarrollando la expresión de (4) y sustituyendo la ecuación anterior en el momento total:

$$\begin{bmatrix} lU_2 \\ lU_3 \\ U_4 \end{bmatrix} = J \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times J \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} x \begin{bmatrix} 0 \\ 0 \\ J_r\Omega_r \end{bmatrix} \tag{27}$$

Donde J es la matriz diagonal de inercia, la que contiene los momentos de inercia I_{xx}, I_{yy} y I_{zz} en la diagonal principal. Operando términos se obtienen las siguientes ecuaciones:

$$\ddot{\phi} = \dot{\phi}\dot{\psi}\frac{I_y - I_z}{I_x} - \frac{J_r}{I_x}\dot{\theta}\Omega + \frac{1}{I_x}U_2 \tag{28}$$

$$\ddot{\theta} = \dot{\theta}\dot{\psi}\frac{I_z - I_x}{I_y} - \frac{J_r}{I_y}\dot{\theta}\Omega + \frac{1}{I_y}U_3 \tag{29}$$

$$\ddot{\psi} = \dot{\theta}\dot{\phi}\frac{I_x - I_y}{I_z} + \frac{1}{I_z}U_4 \tag{30}$$

Por otro lado, para las posiciones XYZ se trabaja con la fuerza total sobre el dron en (14). Al considerar que la fuerza no debida a la gravedad se iguala con la entrada de control y desarrollando (13):

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ -U_1 \end{bmatrix} \tag{31}$$

Desarrollando ecuación matricial se tienen las siguientes ecuaciones del movimiento cartesiano:

$$\ddot{x} = \frac{cos(\phi)sin(\theta)cos(\psi) + sin(\theta)sin(\psi)}{m}U_1 \tag{32}$$

$$\ddot{y} = \frac{cos(\phi)sin(\theta)sin(\psi) - sin(\theta)cos(\psi)}{m}U_1 \tag{33}$$

$$\ddot{z} = -g + \frac{cos(\phi)cos(\theta)}{m}U_1 \tag{34}$$

VI. ALGORITMOS DE CONTROL

Una vez se ha descrito la dinámica de un UAV se pueden diseñar algoritmos para su control. En este apartado se van a explicar algunos de los algoritmos que se usan en los robots aéreos no tripulados, desde el control a algunos ejemplos de algoritmos de mapeado y navegación basados en SLAM. En ambos casos se verán los resultados de cada unos de los algoritmos presentados, basados en investigaciones científicas.

En esta campo existen gran variedad de controladores (LQR, Linearización de la retroalimentación, Controladores Adaptativos Inteligentes mediante lógica difusa, etc) aunque solo se va a profundizar en las técnicas basadas en PID (Proportional Integral Derivative Control), SMC (Sliding Model Control) y Backstepping. [5]

A. *PID*

El control PID es el más extendido en la literatura, por su sencillez y la capacidad de modificar sus parámetros fácilmente y con resultados comprobables y más o menos predecibles. Aun así, se trata de un modelo lineal aplicado a un sistema no lineal provoca ciertas inestabilidades a la hora del control, lo que limita su actuación. A continuación se muestra un esquema donde se observa el control del dron, donde a partir del error entre una consigna y la magnitud de real se calculan las señales de control al robot. Primero, la ganancia proporcional por el error, luego se calcula la derivada del error multiplicada por una ganancia derivativa. Finalmente, se tiene la parte integral, donde se calcula integral del error y se multiplica por una ganancia. Este esquema se aplica a cada uno de los robots, siendo las consignas de velocidad o de aceleración para los motores:

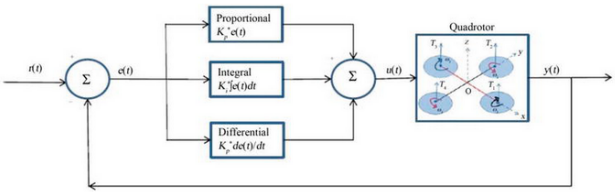

Fig. 6. Esquema PID

Este esquema se ha probado que es robusto solo ante ciertas situaciones, ya que se tienen gran cantidad de perturbaciones que no llegan a compensarse por el control PID. A la hora de mantenerse en el aire, pero el control se hace más difícil. [5] [6]

Algunos algoritmos de control intentan solucionar el problema de la no linearidad del sistema haciendo que la realimentación del error sea lineal y aplicar técnicas como los controladores PID, aunque este tipo de aproximaciones sufren del mismo problema que el controlador previo: son altamente inestables frente a perturbaciones externas, aunque en entornos ideales presenta mejor seguimiento de órdenes que un control PID puro.

B. *SMC*

El algoritmo de SMC se basa en dos partes; una se encarga de mantener el error cerca de la consigna (como un controlador PID tradicional). Este horizonte es el que se conoce como superficie de deslizamiento o *sliding surface*. La otra parte del control solo actúa cuando se detecta que se está dentro de los márgenes de la superficie, manteniendo la salida del robot dentro de márgenes aceptables a la vez que monitoriza la dinámica del robot. [3] [4] [7]

Como ya se vio en la dinámica, se tienen cuatro variables de control para controlar la altitud (U_1) y los ángulos de giro (U_2, U_3, U_4). Teniendo esto en cuenta, se puede definir la fórmula de la superficie o ley de control:

$$s = \dot{e} + \lambda e \tag{35}$$

Donde λ es un parámetro a ajustar.

Como se mencionó anteriormente, la ley de control tiene dos partes, por lo que para cada una de las variables a controlar se tiene:

$$U(t) = u_{eq}(t) + u_D(t) \tag{36}$$

Donde la primera parte es la función de la variable a controlar y la segunda incluye un conmutador para variar los lados de la superficie de deslizamiento:

$$u_D(t) = k_D \Theta(s(t)) \tag{37}$$

Donde la k es un factor ajustable y la función Θ es una función no linear. De esta manera, se puede escribir la siguiente expresión para evitar el *chattering effect* [8], un fenómeno causado por imprecisiones en el modelo que provocan variaciones indeseadas en amplitud y frecuencia.

$$u_D(t) = k_D \frac{s(t)}{|s(t)| + \delta} \tag{38}$$

El parámetro δ es el que evita o mitiga la aparición del efecto *chattering*.

Hasta aquí se tiene la expresión no lineal. Sin embargo, para hallar la otra función se hacen una serie de suposiciones, empezando por la definición del error:

$$e = z_d - z \tag{39}$$

Donde z_d es la consigna de control y z es la variable real a controlar. Durante el desarrollo se van a mostrar los cálculos con la altitud z, pero es aplicable a cualquier otra de las variables, como puede ser la oritentación o el desplazamiento en los otros ejes del sistema de referencia.

Al sustituir en la ecuación se tiene:

$$s = (\dot{z_d} - \dot{z}) + \lambda(z_d - z) \tag{40}$$

La condición de deslizamiento implica que la velocidad es nula, por lo que $\dot{s} = 0$. De esta manera se modifica la ecuación anterior:

$$\dot{s} = (\ddot{z_d} - \ddot{z}) + \lambda(\dot{z_d} - \dot{z}) \tag{41}$$

Sustituyendo las ecuaciones de la dinámica en la ley de control se tiene:

$$\dot{s} = (\ddot{z_d} + g - \frac{cos(\phi)cos(\psi)}{m} U_1) + \lambda(\dot{z_d} - \dot{z}) \tag{42}$$

Al estar dentro de la condición de deslizamiento, es decir, en el equilibrio, se puede considerar que U_1 es u_{eq}. Así, despejando esta variable se puede conseguir la primera parte de la fórmula de control:

$$u_{eq} = [g + \lambda(\dot{z_d} - \dot{z}) + \ddot{z_d}] \frac{m}{cos(\phi)cos(\psi)} \tag{43}$$

Finalmente, la función queda de la siguiente manera:

$$U_1 = [g + \lambda(\dot{z_d} - \dot{z}) + \ddot{z_d}]\frac{m}{cos(\phi)cos(\psi)} + k_D\Theta(s(t)) \quad (44)$$

Así, solo queda obtener la función u_D para mantener la variable controlada dentro de la superficie definida, esto es, en el equilibrio. Para ello, se define una función de Lyapunov [9]. Una función de Lyapunov es una función que cumple el criterio del mismo autor, que establece que una función escalar que se define positiva en un espacio continuo, si su derivada primera es definida negativa, se garantiza la estabilidad de los ceros de la misma. Así, la función elegida junto con su derivada resulta:

$$V = \frac{1}{2}s^2 > 0 \quad (45)$$

$$\dot{V} = s\dot{s} < 0 \quad (46)$$

Teniendo en cuenta esto se tiene que $K_d > 0$ durante todo el tiempo. Así, completando la ecuación (44) sustituyendo la ecuación (38):

$$U_1 = [g + \lambda(\dot{z_d} - \dot{z}) + \ddot{z_d}]\frac{m}{cos(\phi)cos(\psi)} + k_D\frac{s(t)}{|s(t)|+\delta} \quad (47)$$

El mismo procedimiento se siguió para todas las demás variables.

Así, se pueden pasar las señales de control a los rotores a través de las ecuaciones descritas en (22), (23), (24) y (25) de forma matricial.

En este caso, se tiene un control PD del algoritmo SMC, donde la k_D representaría la ganancia proporcional y la λ la ganancia derivativa.

Este control incorpora componentes no lineales y tiene en cuenta la dinámica del sistema, por lo que resulta más fiable que un controlador PID tradicional. El comportamiento puede ser regulado ajustando las diferentes ganancias, siendo esta una posible respuesta del sistema extraída de los resultados de [4]:

En la figura se puede ver como el robot es capaz de seguir comandos de posición en altitud con bajo error y oscilación. Algunas inestabilidades se pueden dar por perturbaciones externas, aunque el sistema es capaz de contrarrestarlas. El control en posición es posible gracias a sensores de altitud para la coordenada z, aunque en los otros ejes se puede conseguir retroalimentación de la doble integración de la aceleración o de sensores internos como IMUs, así como de localizadores externos.

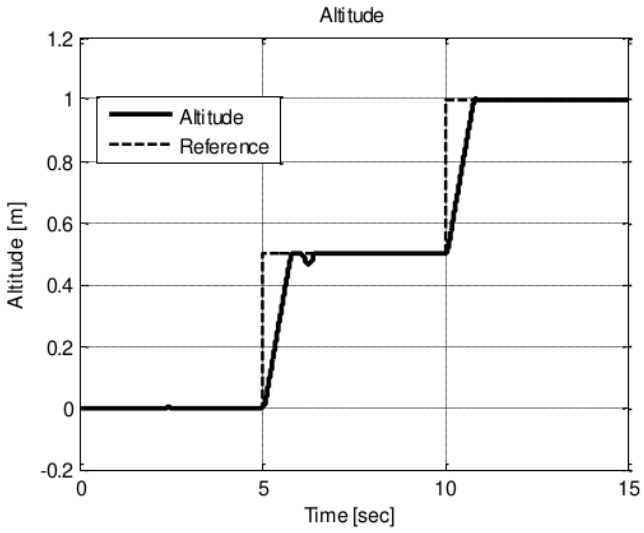

Fig. 7. Gráfica del despegue de un dron de cuatro hélices con control PD-SMC

C. *Backstepping*

El principio de este tipo de control es el de dividir el funcionamiento del sistema en varios módulos independientes. Este tipo de control se hace para aplicar las teorías de Lyapunov de modelos lineales a modelos no lineales sin tener que pasar por la linearización de la retroalimentación. [7] [10] [11]

Cabe destacar que para realizar los cálculos de este algoritmo de control se requiere hace cambios de variable y variar el espacio de estados que se había definido anteriormente:

$$\dot{x_1} = x_2 \quad (48)$$

$$\dot{x_2} = a_1x_4x_6 + a_3\Omega x_4 + b_1U_2 \quad (49)$$

$$\dot{x_3} = x_4 \quad (50)$$

$$\dot{x_4} = a_4x_2x_6 + a_6\Omega x_2 + b_2U_3 \quad (51)$$

$$\dot{x_5} = x_6 \quad (52)$$

$$\dot{x_6} = a_7x_2x_4 + b_3U_4 \quad (53)$$

$$\dot{x_7} = x_8 \quad (54)$$

$$\dot{x_8} = \frac{cos(x_1)cos(x_2)}{m}U_1 - g \quad (55)$$

$$\dot{x_9} = x_{10} \quad (56)$$

$$\dot{x_{10}} = U_y\frac{U_1}{m} \quad (57)$$

$$\dot{x_{11}} = x_{12} \quad (58)$$

$$\dot{x_{12}} = U_x\frac{U_1}{m} \quad (59)$$

Donde cada x_* indica una coordenada a controlar o ánugulo *roll, pitch, yaw* de orientación. Con $\Omega = \Omega_2 + \Omega_4 - \Omega 1 - \Omega_3$.

$$a_1 = \frac{I_y - I_z}{I_x},\ a_3 = \frac{J_r}{I_x},\ a_4 = \frac{I_z - I_x}{I_y},\ a_6 = \frac{J_r}{I_y}, \quad (60)$$

$$a_7 = \frac{I_x - I_y}{I_z},\ b_1 = \frac{1}{I_x},\ b_2 = \frac{d}{I_y},\ b_3 = \frac{d}{I_z} \quad (61)$$

Denotado I_* los momentos de inercia y J_r la inercia debida a los motores.

A la hora de la formulación se parte de la misma función de Lyapunov presentada anteriormente, solo que en este caso se aplica el error en vez de una ley de control:

$$V_1 = \frac{1}{2}e_1^2 > 0 \quad (62)$$

$$\dot{V_1} = e_1\dot{e_1} < 0 \quad (63)$$

Sabiendo que $\dot{e_1} = \dot{x_{d1}} - \dot{x_1}$ se puede establecer la siguiente relación; $\dot{e_1} = -Ke_1$, por lo que se puede obtener el término $x_{d2} = \dot{x_{d1}} + K_1e_1$. Teniendo en cuenta que la ecuación del error en velocidad es $e_2 = x_{d2} - x_2$, junto con la ecuación (49) del sistema en espacio de estados y aplicando $V_2 = V_1 + \frac{1}{2}e_2^2$ como función de Lyapunov se obtiene:

$$\begin{aligned}
\dot{V_2} &= \dot{V_1} + e_2\dot{e_2} \\
&= e_1\dot{e_1} + e_2\dot{e_2} \\
&= e_2(\dot{x_{d1}} - x_2) + e_2(x_{d2} - \dot{x_2}) \\
&= e_1(x_{d1} - (\dot{x_{d1}} - e_2)) \\
&\quad + e_2(x_{d2} - (x_4x_6a_1 - x_4\Omega a_2 + b_1U_2)) \\
&= e_1(x_{d1} - \dot{x_{d2}}) + e_1e_2 \\
&\quad + e_2(x_{d2} - (x_4x_6a_1 - x_4\Omega a_2 + b_1U_2)) \\
&= -K1e_1^2 + e_2(e_1 + \dot{x_{d2}} - x_4x_6a_1 + x_4\Omega a_2 - b_1U_2) \\
&= e_1 + \dot{x_{d2}} - x_4x_6a_1 - b_1U \\
&= -K_2e_2
\end{aligned} \quad (64)$$

Así, se obtiene la ley de control para la primera variable:

$$U_2 = \frac{1}{2}[e_1 - K_1x_2 - x_4x_6a_1 + x_4\Omega a_2 + K_2e_2] \quad (65)$$

El mismo procedimiento se realiza para cada una de las otras variables del espacio de estado siguiendo los cambios de variable y los estados descritos anteriormente. Juntando todas las variables a controlar, quedaría el siguiente diagrama de flujo en el control:

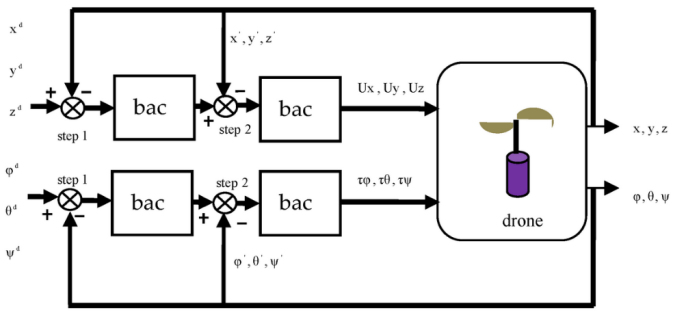

Fig. 8. Esquema de control para Backsteping

Aquí se puede ver como se calculan los errores tanto en velocidad como en posición para luego generar los comandos y señales de control que definen el comportamiento del robot mientras vuela.

Este modelo de control es similar al SMC en cuanto a que los dos usan el modelo de Lyapunov para lograr la estabilidad asintótica y que en ambos se necesita de la velocidad y la posición para su correcto funcionamiento. Aun así, comparando los distintos resultados que ofrece cada uno de estos algoritmos en [7] se puede comprobar que el control con SMC resulta más rápido y presenta menor oscilación que el realizado con *Backstepping*, aunque ambos son estables y resistentes ante perturbaciones externas.

VII. Movimiento autónomo

Una vez se tienen modelos con los que controlar el dron en el aire y se conocen algunos métodos para realizar un control a bajo nivel del vehículo se puede subir de nivel en la jerarquía de tareas del robot. Así, se pasa de controlar el robot a hacer que este navegue por un espacio conocido de manera autónoma. Esta tarea tiene una complejidad mayor que la que se haría con los robots móviles terrestres ya que se manejan muchos más grados de libertad en los drones (3 para los terrestres y 6 para los aéreos o marinos).

Para realizar este tipo de control de más alto nivel se necesita de algunos algoritmos a parte de los de control de rotores. Un ejemplo de estos sería SLAM (*Autonomous Localization And Mapping*), que necesita de una representación probabilística de los datos y del estado interno del propio robot en cada instante de tiempo.

A. *Estimación del estado*

Antes de explicar los algoritmos de mapeado y localización se deben definir los sensores que tiene un supuesto robot aéreo y como usar la odometría del dron para estimar la posición en el espacio [12]. Además de especificarlos también es importante definir como se computan los datos de los sensores para obtener información de la posición; aunque el dron se controle mediante velocidad, a la hora de construir el mapa y localizarlo es más útil integrar la velocidad y la aceleración para obtener la posición del dron en el mundo. Así, los sensores que comúnmente se aplican en los robots aéreos son:

- **Giroscopios y acelerómetros**: son dispositivos que miden la velocidad angular y la aceleración respectivamente. Además, con el primer elemento se puede calcular la orientación absoluta del dron mediante la integración de la velocidad angular, aunque esto provoca una deriva en el resultado.
- **Cámaras**: los drones destinados a navegación autónoma suelen disponer de una cámara, tanto para el mapeado como para la posterior navegación y localización. También se usan para corregir desviaciones en el cálculo de la posición

- **Altímetro**: se usan dispositivos para determinar la altura del dron respecto al suelo. Suele tratarse de sensores de ultrasonidos enfocados en la dirección negativa del eje Z del sistema de referencia del robot según la Figura IV.

A la hora de combinar todas estas mediciones, los equipos suelen tener algoritmos para obtener buenos resultados a pesar de las perturbaciones o el ruido de los sensores, fusionando y filtrando las magnitudes. Uno de los métodos usados es la compensación de la aceleración de los acelerómetros en los aterrizajes; al aterrizar se produce una amortiguación en el acelerómetro, un movimiento que no se da durante el vuelo, por lo que se debe estimar esta corrección cuando el dron llegue a tierra.

Por otro lado, la deriva en la integración de la velocidad de los giroscopios se suele corregir con algoritmos de *Visual Odometry* [13], donde se estiman las velocidades de la cámara respecto a lo que está viendo. Se tienen varias aproximaciones; calcular la velocidad a partir del *Optical Flow* [14], la variación de un grupo de píxeles respecto de una toma de la cámara a la siguiente. También se pueden extraer los puntos clave de dos o más imágenes consecutivas y estimar la velocidad en el tiempo. Con esto se produciría una especie de estéreo.

Aun así, estos algoritmos son muy lentos para la actuación en tiempo real del robot, por lo que usan los sensores propioceptivos del robot mencionados (acelerómetros y giroscopios) como entradas a un modelo dinámico. De esta manera, fusionando el modelo dinámico y la información de la cámara se reduce la derivación de la integración de la aceleración y se filtra la odometría visual de la cámara.

De esta manera, se puede crear un vector de medidas, donde guardar cada uno de los estados del robot:

$$x = [p^W \quad v^W \quad a^W \quad q^W \quad h^W]^T \tag{66}$$

Donde x es el vector de medidas, p^W es la posición del robot, v^W es la velocidad, a^W es la aceleración, q^W es la orientación en ángulos de **roll, pitch, yaw** y h^W es la estimación de la altitud con el altímetro del dron. Esta información se obtiene de los sensores mencionados previamente. Con estos datos se puede calcular la posición siguiendo la siguiente ecuación:

$$p_t = p_{t-1} + v_t \times \Delta_t + a_t \times 0.5\Delta_t^2 \tag{67}$$

Donde Δ_t es el intervalo de tiempo entre dos medidas. Hasta aquí, se tiene una manera de estimar la posición a partir de las velocidades y aceleraciones del dron. Es usual utilizar un modelo basado en EKF o cualquier otro modelo.

B. SLAM

El mapa es la representación que tiene el robot del entorno en el que esté y es el primer paso a la hora de programar una tarea con un robot móvil. Una técnica ampliamente utilizada en todo tipo de robots móviles es el SLAM, la localización y mapeado simultáneos del entorno, lo que permite crear un mapa del entorno.

En los drones, una variante de esta técnica podría ser el Visual-SLAM [15], que usa reconocimiento de imágenes y detección de puntos clave para la localización del robot. Uno de los algoritmos más utilizados es el ORB-SLAM [16] en el que se usa ORB como detector de puntos clave, pues es un elemento robusto e invariante a rotaciones, lo que resulta ideal para este tipo de actividades.

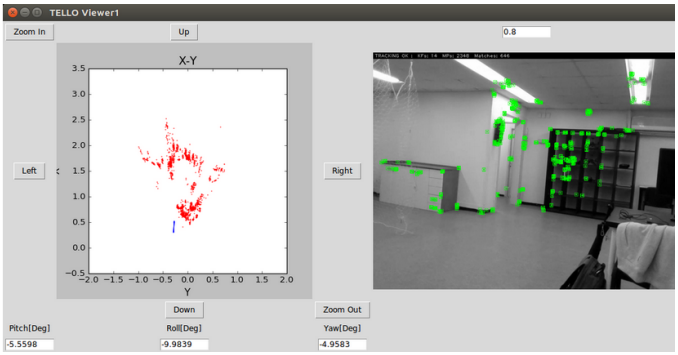

Fig. 9. Salida de un algoritmo de detección ORB

En la imagen anterior se puede ver la salida de un algoritmo de SLAM - ORB implementado en ROS [17] donde se puede ver la detección de los puntos clave. La técnica de Visual SLAM con *landmarks* que se propone se basa en el concepto de; medir los puntos de interés y identificarlos, usarlos para localizar el robot. Luego se proyecta el movimiento (ya sea por la información de la odometría o por las órdenes de control), aumentando la incertidumbre. Finalmente se actualiza el estado. Cabe destacar que los dos primeros pasos se pueden permutar.

Al basar SLAM en puntos clave es útil utilizar la aproximación a la técnica usando Filtros de Kalman, en este caso el Filtro de Kalman Extendido (EKF) [18]. Este filtro tiene información de las variables del sistema gracias a la expresión (66), aunque se suele usar solo la posición en el mapa, tanto del robot como de los puntos clave de la siguiente manera:

$$x = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \\ \dots \\ m_{ix} \\ m_{iy} \\ m_{iz} \\ m_{i\phi} \\ m_{i\theta} \\ m_{i\psi} \\ \dots \end{bmatrix} \tag{68}$$

Donde las tres primeras variables indican la posición en el espacio cartesiano del robot, las tres siguientes son la orientación y las siguientes son las coordenadas de i-ésimo punto clave. En cada uno de los elementos del vector de estados se tienen en cuenta todos los grados de libertad tridimensionales. Así, se puede construir la matriz de covarianza, necesaria para tener una representación de la incertdumbre en la posición de cada elemento del sistema.

$$P = \begin{bmatrix} P_r & P_{rm_1} & ... & ... & P_{rm_N} \\ P_{rm_1} & P_{m_1} & P_{m_1m_2} & ... & P_{m_1m_N} \\ ... & ... & ... & ... & ... \\ P_{rm_N} & P_{m_1m_N} & ... & ... & P_{m_N} \end{bmatrix} \tag{69}$$

Donde P es la matriz de covarianza del sistema, P_r es la submatriz de covarianza del robot, P_{mi} son la submatrices de covarianza del i - ésimo *landmark* y P_{rmi} son las matrices de covarianza de los *landmarks* respecto al robot. Cada una de las submatrices es de dimensión 6×6, una por cada grado de libertad en el espacio.

Así, se pueden diferenciar las siguientes etapas:

- **Predicción del estado**: basándose en el modelo de movimiento y en las señales de control se establece una posición virtual del robot en el espacio, calculando la incertidumbre *a priori* con las siguientes expresiones:

$$P_r = F_x P_r F_x^T + F_u U F_u^T \tag{70}$$

$$P_{rmi} = F_x P_{rmi} \tag{71}$$

  En la primera se calcula la covarianza del robot teniendo en cuenta lo que debería ver F_x y la odometría U y el modelo de movimiento T_u. Luego, la matriz de covarianza del mapa se actualiza con lo que debería ver el vehículo y la matriz anterior, pues solo se computa la información relativa al *landmark*.

- **Medición**: una vez se tiene la predicción, se modelan las medidas de los sensores del dron a través de sus sensores, comparándolas con las medidas predichas.

$$\tilde{z_k} = [z_1 \quad z_2] \tag{72}$$

  Donde $\tilde{z_k}$ es la medida real de los sensores en la k - ésima variable a controlar y denotando Q_k como la covarianza de dicha medida, que indica el ruido o la precisión que tiene. Así, suponiendo que las medidas predichas se expresan como z_k se puede obtener la innovación, como varía la medida predicha de la real y su matriz de covarianza.

$$s_k^{ij} = \tilde{z_k}^j - z_k^i \tag{73}$$

$$S_k^{ij} = R_k^j + H^i P_k H^{iT} \tag{74}$$

  Donde s_k^{ij} es la innovación entre las variables de posición de cada *landmark* y S_k^{ij} es la matriz de covarianza. Así se tiene una estimación de la incertidumbre de los puntos clave entre ellos y individualmente.

- **Corrección**: una vez se han combinado las medidas predichas con las reales se procede a corregir la información en el modelo en base a las siguientes expresiones.

$$K_k = P_k H^T S_k^- 1 \tag{75}$$

$$x_k = x_k + K_k s_k \tag{76}$$

$$P_k = (I - K_k H) P_k \tag{77}$$

  Donde la H es la matriz hessiana de la covarianza y P_k es la k - ésima submatriz de covarianza. La ecuación (77) actualiza la variable x_k del vector de estados con la posición anterior y la innovación s_k ponderada por la covarianza K_k, obtenida de multiplicar la covarianza, su derivada (hessiana) y la de la innovación. Finalmente, se calcula la nueva submatriz de covarianza final.

- **Adición de nuevos *landmarks***: el robot al inicio no puede saber todos los puntos clave que hay en el espacio, los detecta a medida que se va moviendo por el entorno. Por eso, la matriz de covarianza P va aumentando conforme avanza la ejecución. Así, se calcula una nueva submatriz de covarianza para el nuevo punto.
  Teniendo:

$$m_{N+1} = g(x_r, z_j) \tag{78}$$

$$G_r = \frac{\partial g}{\partial x_r}, \quad G_z = \frac{\partial g}{\partial z_j} \tag{79}$$

  Como la posición en el espacio del nuevo *landmark* y denotando G la covarianza de su posición (con las matrices G_r y G_z), se pueden obtener las siguientes matrices a partir de la nueva información y la que ya se tiene:

$$P_{mN+1} = G_r P_r G_r^T + G_z R_j G_z^T \tag{80}$$

$$P_{m_{N+1}m_i} = G_r P_{rm_i} \tag{81}$$

$$P_{m_{N+1}r} = G_r P_r \tag{82}$$

  Donde primero se calcula la covarianza del *landmark*y luego se obtienen las cruzadas entre los demás elementos y finalmente la del robot.

Como se puede apreciar este algoritmo presenta multitud de operaciones matriciales, por lo que con el aumento de *landmarks* en el entorno, se produce un crecimiento cuadrático en la dimensión de las matrices, así como en el tiempo que requiere el cálculo de covarianzas. Esto es provocado por el hecho de que al detectar un solo objeto, se actualizan todos los valores de P. Una solución a este problema sería la de discernir que valores de la matriz de covarianza no son útiles para la localización y obviarlos o anularlos.

Por otro lado, se podría reducir el número de *landmarks* que se tienen en cuenta, restringiendo el detector o manteniendo densidades de puntos por el mapa, de manera que se tienen solo los más significativos por zonas. Siguiendo con esta aproximación, se podría estudiar cuales puntos son los que reducen la incertidumbre al ser detectados, dándoles prioridad.

Un fenómeno a tener en cuenta es el cierre del bucle; al moverse, el dron aumenta su incertidumbre y los nuevos *landmarks* la reducen pero no lo suficiente. Así, el robot transfiere su error al de los *landmarks*. Esta incertidumbre se reduce al volver al punto inicial, ya que se encuentra en una zona con bajo error, con innovación es alta, corrigiendo las covarianzas de los demás elementos.

VIII. Ejemplos de aplicaciones

Para ilustrar más a fondo el comportamiento de los algoritmos explicados anteriormente, se exponen a continuación una serie de ejemplos prácticos de drones que hacen uso de estos métodos, así como aplicaciones reales que hacen uso de drones en la actualidad.

A. *Ejemplos prácticos*

En este primer apartado se muestran una serie de ejemplos prácticos útiles en educación e investigación, para observar el funcionamiento de los algoritmos.

*1) **Control PID de un cuadricóptero**:* Este primer ejemplo desarrolado por Brian Wade [19], sirve como ejemplo sencillo para observar el comportamiento que tiene el control PID en un dron cuadricóptero. Este proyecto se trata de una simulación de la dinámica del vehículo a lo largo de una trayectoria, en el que dos controladores PID ajustan la velocidad y habilitan el control de la posición y orientación.

En la simulación, inicialmente se establen la posición y la velocidad referidas al origen del sistema inercial, además de los ángulos *roll, pitch, yaw* del dron. Además, cabe destacar que la simulación también admite una perturbación inicial aleatoria sobre cada ángulo de Euler, y se establece una magnitud máxima (º/s) de perturbación permitida que puede ocurrir en cada ángulo. Como se ha mencionado anteriormente, el cuadricóptero utiliza dos controladores PID para controlar las cuatro velocidades del motor. Un controlador PID controla la posición y genera un conjunto de ángulos de Euler ya que el dron debe maniobrar cambiando la orientación de su cuerpo. El segundo es para el control de la orientación y utiliza los ángulos de Euler objetivo para generar nuevas velocidades de motor deseadas para cada uno de los cuatro motores del cuadricóptero que accionan sus hélices. Al aplicar este control se obtienen los siguientes resultados:

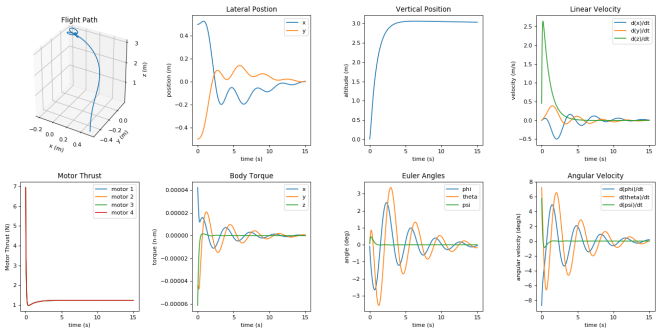

Fig. 10. Resultados del control PID

En la primera fila de esta figura se observa la trayectoria 3D descrita durante el vuelo, el movimiento lateral (x,y), movimiento vertical (z) y la velocidad lineal a lo largo del tiempo para cada uno de los tres ejes del sistema de referencia interno. Mientras que en la segunda fila se observan las gráficas que describen el empuje de cada uno de los cuatro motores, el par sufrido en cada eje del dron, la variación de los ángulos de Euler y la velocidad angular a lo largo de la trayectoria. Es en estas últimas gráficas en las cuales se aprecia el efecto del controlador PID, generando la acción de control necesaria para anular el error con respecto a las consignas.

*2) **Localización y mapeado en ROS**:* Otro proyecto interesante para ilustrar el funcionamiento de una aplicación de un dron con movimiento autónomo es [20], el cual se trata de un porte a ROS Noetic del dron *hector_quadcopter* [21], un modelo bastante extendido en ROS frecuentemente utilizado como dron de pruebas de algoritmos. Este stack de ROS incluye paquetes relacionados con el modelaje, control y simulación de UAVs cuadricópteros. Éste incluye varios modelos URDF genéricos con diferentes sensores, paquetes que contienen archivos .launch y dependencias necesarias para la simulación en gazebo y paquetes para la teleoperación del dron.

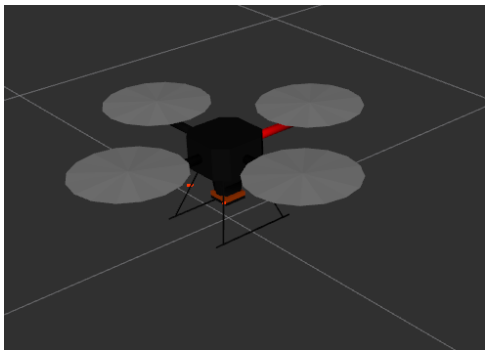

Fig. 11. Hector_quadcopter

Además de esto ROS dispone tutoriales para realizar SLAM con UAVs cuadricópteros haciendo uso de este stack, cuyos paquetes proporcionan la pose 6DOF completa de un robot o plataforma. El equipo Hector Darmstadt utiliza esto para estimar la pose 6D completa del robot dentro del bucle en

tiempo real del dron e incluso para estimar la posición, velocidad y altitud de un pequeño avión.

El paquete central proporciona un modelo de sistema para la estimación genérica de la pose de un cuerpo rígido 6DOF basado en entradas IMU puras (tasas y aceleraciones), que puede ser especializado en función del robot y para valores de entrada adicionales. Como modelos de medición, el paquete proporciona mediciones directas y barométricas de altura, posición y velocidad GPS, sensores de campo magnético como referencia de rumbo y una medición genérica de pose y torsión para fusionar mensajes *nav_msgs/Odometry* de fuentes arbitrarias (por ejemplo, odometría de rueda o SLAM). En segundo plano, un Filtro de Kalman Extendido basado en la Biblioteca de Filtros Bayesianos es responsable de fusionar toda la información en una estimación de estado consistente y, además, realiza un seguimiento de qué variables de estado son observables y cuáles no.

Realizando pruebas con el paquete *hector_quadotor_demo*, se ha realizado el mapeado de la simulación al aire libre, obteniendo el siguiente resultado:

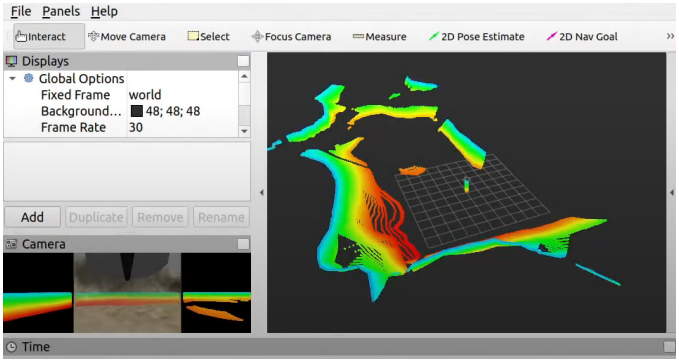

Fig. 12. Mapeado con el hector_quadcopter

Este mapeado se realiza utilizando el paquete *hector_mapping* del stack de *hector_slam*. Cabe destacar que la estimación de la actitud proviene de gazebo, pero la localización 2D y el mapeado no utilizan información *'ground truth'* del terreno. Esto también se puede comprobar volando demasiado alto, lo que provoca un fallo de SLAM. La trayectoria 3D del robot es rastreada y visualizada en rviz usando el paquete hector_trajectory_server. Un mapa que incluya la trayectoria se puede guardar en un archivo GeoTiff, mediante:

```
rostopic pub syscommand
std_msgs/String "savegeotiff"
```

B. *Ejemplos de aplicaciones reales*

A continuación se exponen diversos ejemplos actuales que tienen los drones en la sociedad.

1) ***Agricultura****:* Los drones se utilizan cada vez más en la agricultura para realizar tareas como el mapeo del campo, la identificación de plagas y enfermedades, y el riego preciso. Estos drones pueden volar sobre los campos y recopilar imágenes de alta resolución que se utilizan para analizar la salud de las plantas y determinar las áreas que necesitan más agua o fertilizante. Esto permite a los agricultores tomar decisiones informadas sobre cómo cultivar sus cultivos y aumentar la eficiencia de sus operaciones.

2) ***Paquetería****:* Con el surgimiento de drones con capacidad de transportar paquetes, varias compañías están experimentando con nuevas formas de usar esta tecnología para entregar paquetes a clientes de manera más rápida y eficiente. En lugar de tener que depender de vehículos terrestres para entregar paquetes, los drones pueden volar directamente a las direcciones de los clientes y dejar los paquetes en sus patios traseros. Esto puede reducir significativamente los tiempos de entrega y mejorar la eficiencia de la entrega de paquetes.

3) ***Inspección de infraestructura****:* También se están utilizando cada vez más estos UAVs en la inspección de infraestructura, como líneas de energía, puentes, edificios y presas. Estos drones pueden volar cerca de estas estructuras y tomar imágenes detalladas y vídeos que se utilizan para detectar defectos y problemas potenciales. Esto permite a los ingenieros y técnicos inspeccionar estas estructuras de manera más rápida y eficiente, lo que ayuda a reducir el riesgo de fallos y accidentes.

4) ***Salvamento y búsqueda****:* Desde hace años ya se utilizan estos dispositivos en operaciones de salvamento y búsqueda para ayudar a localizar a personas desaparecidas o en peligro en áreas difíciles de acceder. Estos drones pueden volar sobre terrenos montañosos, bosques, y zonas costeras, y utilizar cámaras y sensores para buscar señales de vida. Además los Drones equipados con termocámaras para detectar señales de calor, pueden ayudar en búsquedas nocturnas o en zonas de difícil acceso. Esto puede ayudar a los equipos de rescate a encontrar a las personas más rápidamente y salvar vidas.

IX. Conclusión

En definitiva, el estudio realizado ha demostrado que los drones son una herramienta tecnológica valiosa para una variedad de aplicaciones, incluyendo el monitoreo de infraestructuras, mapeado, etc. A medida que la tecnología se desarrolla, los drones se están convirtiendo en una herramienta cada vez más accesible y asequible para una variedad de industrias.

Sin embargo, también es importante tener en cuenta que existen desafíos técnicos para el uso de drones, como la autonomía, la estabilidad y la seguridad en vuelo. Es importante seguir investigando y desarrollando nuevas tecnologías para abordar estos desafíos. Además, un aspecto importante que no se ha discutido debido a que no entra en el objetivo de este trabajo, es la integración segura y eficiente de los drones en el espacio aéreo, que requiere el desarrollo de normas y protocolos de seguridad adecuados.

En general, los drones representan una oportunidad emocionante para mejorar la eficiencia y la seguridad en una variedad de campos, y es esencial seguir investigando y desarrollando las tecnologías relacionadas para sacar el máximo provecho de su potencial.

REFERENCIAS

[1] Video presentación del trabajo: https://www.youtube.com/watch?v=lv2h8tIZrfo

[2] Tipos de drones: https://umilesgroup.com/tipos-de-drones/

[3] Elkholy, Heba talla Mohamed Nabil. Dynamic modeling and control of a Quadrotor using linear and nonlinear approaches. (2014): https://fount.aucegypt.edu/cgi/viewcontent.cgi?article=2291&context=etds

[4] Herrera, Marco; Chamorro, William; Gómez, Alejandro; Camacho, Oscar. Sliding Mode Control: An Approach to Control a Quadrotor. (2015): https://www.researchgate.net/publication/279450122_Sliding_Mode_Control_An_Approach_to_Control_a_Quadrotor

[5] P Priya, and S. S. Kamlu, "Robust Control Algorithm for Drones", in Aeronautics - New Advances. London, United Kingdom: IntechOpen, (2022): https://www.intechopen.com/chapters/83163 doi10.5772/intechopen.105966: https://www.intechopen.com/chapters/83163

[6] Adilet Tagaya, Abylkaiyr Omara, Md. Hazrat Ali. (2021). Development of control algorithm for a quadcopter. Nur-Sultan, Kazkahstan.: https://www.sciencedirect.com/science/article/pii/S1877050921000041?ref=cra_js_challenge&fr=RR-1

[7] A. Swarup and Sudhir. Comparison of Quadrotor Performance Using Backstepping and Sliding Mode Control. (2014) : https://www.academia.edu/7413154/Backstepping_slidemode

[8] Vladmin Utkin. Chattering Problem in Sliding Mode Control Systems. (2006): https://www.researchgate.net/publication/271476295_Chattering_Problem_in_Sliding_Mode_Control_Systems

[9] Criterio de Lyapunov: https://mathworld.wolfram.com/LyapunovFunction.html

[10] Saibi, A.; Boushaki, R.; Belaidi, H. Backstepping Control of Drone. (2022): https://www.mdpi.com/2673-4591/14/1/4

[11] Sundarapandian Vaidyanathan, Ahmad Taher Azar. Backstepping Control of Nonlinear Dynamical Systems. (2021): https://www.sciencedirect.com/science/article/pii/B9780128175828000088

[12] Nick Dijkshoorn. Simultaneous localization and mapping with the AR.Drone. (2012): https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f188873bb0ecedf323bbd518e3605afbb7d3bf2f

[13] Visual Odometry: http://www.cs.toronto.edu/ urtasun/courses/CSC2541/03odometry.pdf

[14] Flujo Óprico: https://www.sciencedirect.com/topics/engineering/optical-flow

[15] Mapeado y localización de UAVs: http://icarus.csd.auth.gr/wp-content/uploads/2018/08/4-Localization-and-mapping.pdf

[16] Krul S, Pantos C, Frangulea M, Valente J. Visual SLAM for Indoor Livestock and Farming Using a Small Drone with a Monocular Camera: A Feasibility Study. (2021): https://www.mdpi.com/2504-446X/5/2/41

[17] ORB-SLAM en ROS: https://github.com/tau-adl/Tello-ROS-ORBSLAM

[18] SLAM-EKF: http://oramosp.epizy.com/teaching/201/rob-autonoma/clases/11aEKF-SLAM.pdf

[19] Brian Wade Quadcopter PID Controller simulation in python: https://github.com/brianwade1/quadcopter_with_PID_controller

[20] hector˙quadrotor ported to ROS Noetic Gazebo 11: https://github.com/RAFALAMAO/hector-quadrotor-noetic

[21] ROS hector quadcopter: http://wiki.ros.org/hector_quadrotor